Signal and Systems

Programming Assignment

Team Member 1 : B20AI021 [Mitul Agrawal]

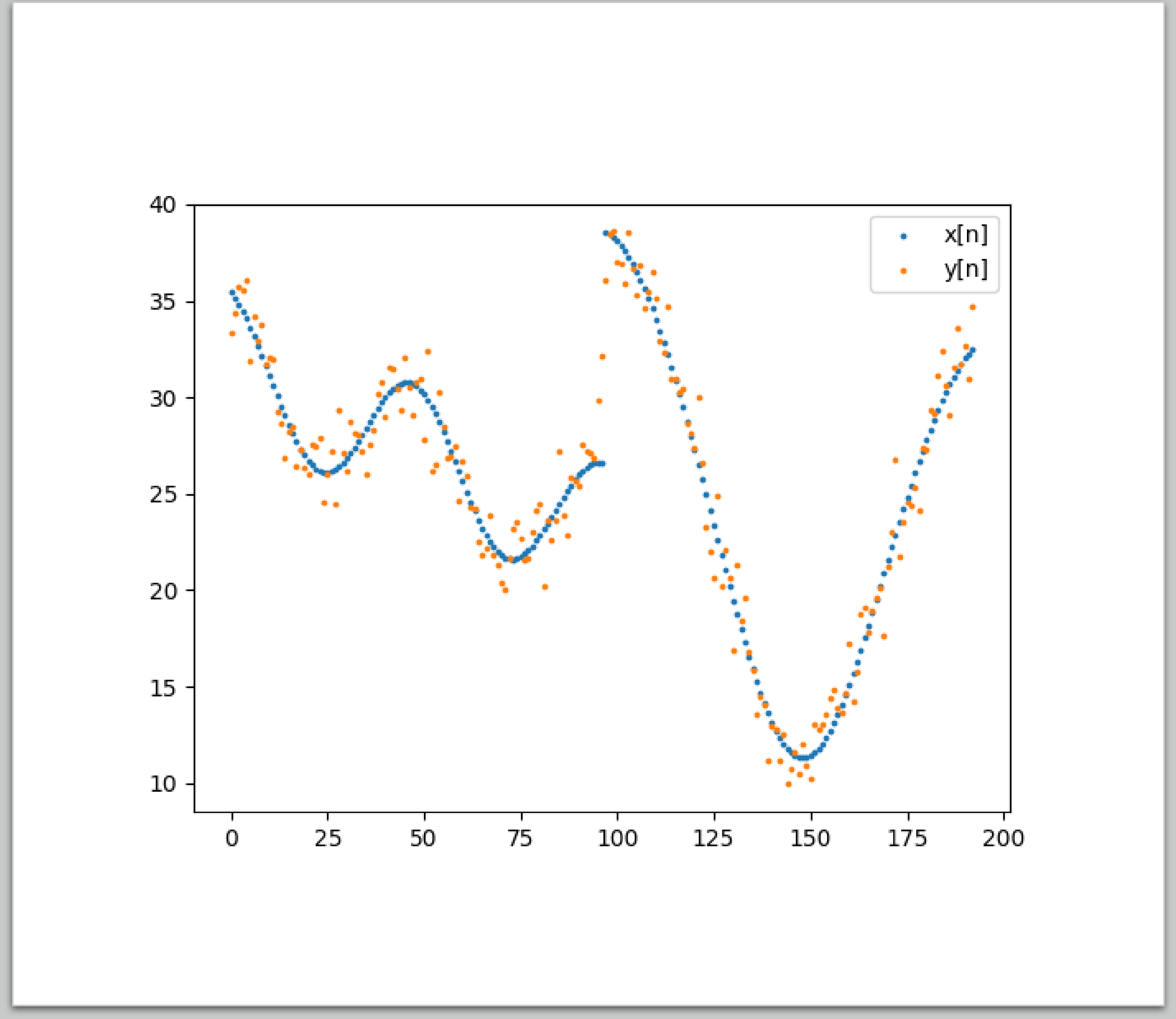
Team Member 2 : B20MT005 [Aman Vashishth]

Aim

To recover a distorted signal using denoising and deblurring

This will enable us to apply the concepts we have learned in the course.

- x : True Values
- y : Distorted Values
- h : Impulse Response
- x1 : Denoise -> Deblur
- x2 : Deblur -> Denoise



Denoising

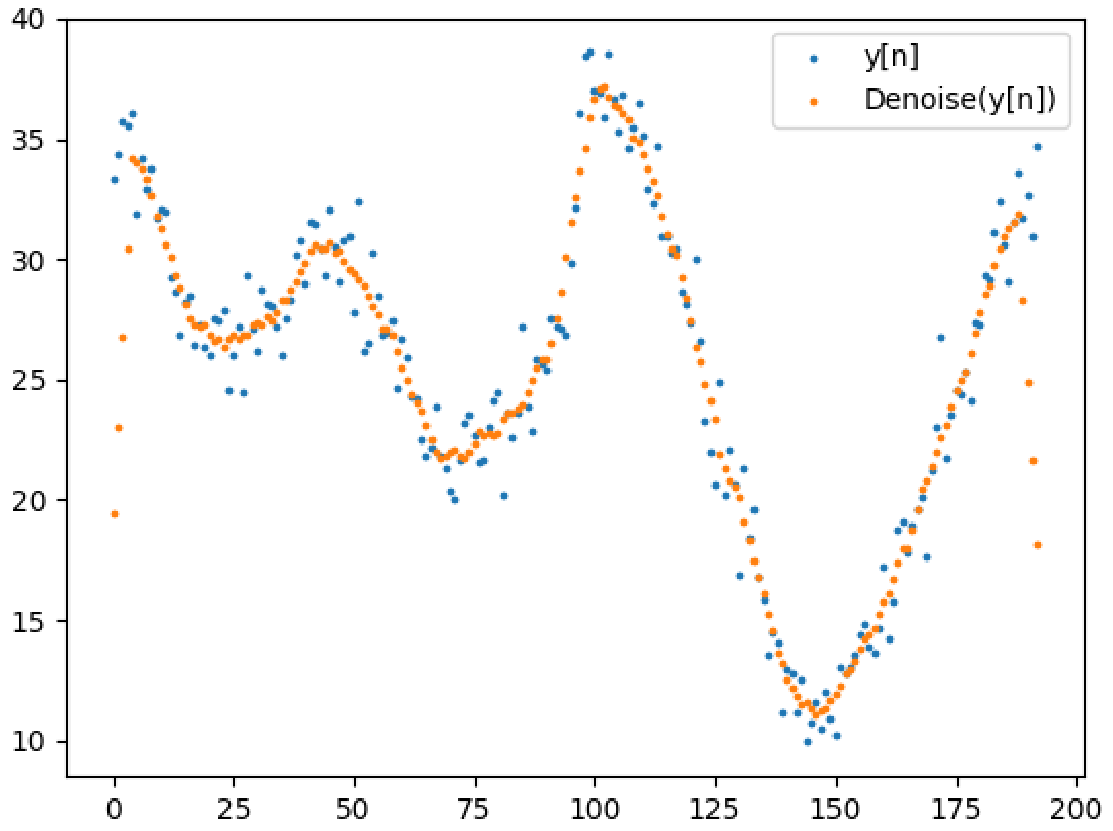
For Denoising we defined a kernel (low pass filter) and took the convolution of y[n] with the given kernel.

Kernel : 1/9 [1,1,1,1,1,1,1,1,1]

This method (kernel used) is basically taking average of surrounding values. The logic behind this is that noise in the values is random, so the random fluctuations below and above will balance each other.

In the graph we can see some values get lower. This happens at the boundaries when the kernel doesn't fully overlap with the data elements during convolution.

Averaging acts as a low pass filter : Noise is caused by rapid changes in the amplitude which can be seen as high frequency components. Averaging helps in reducing the rapid change, so in a way it allows the low frequency components to pass



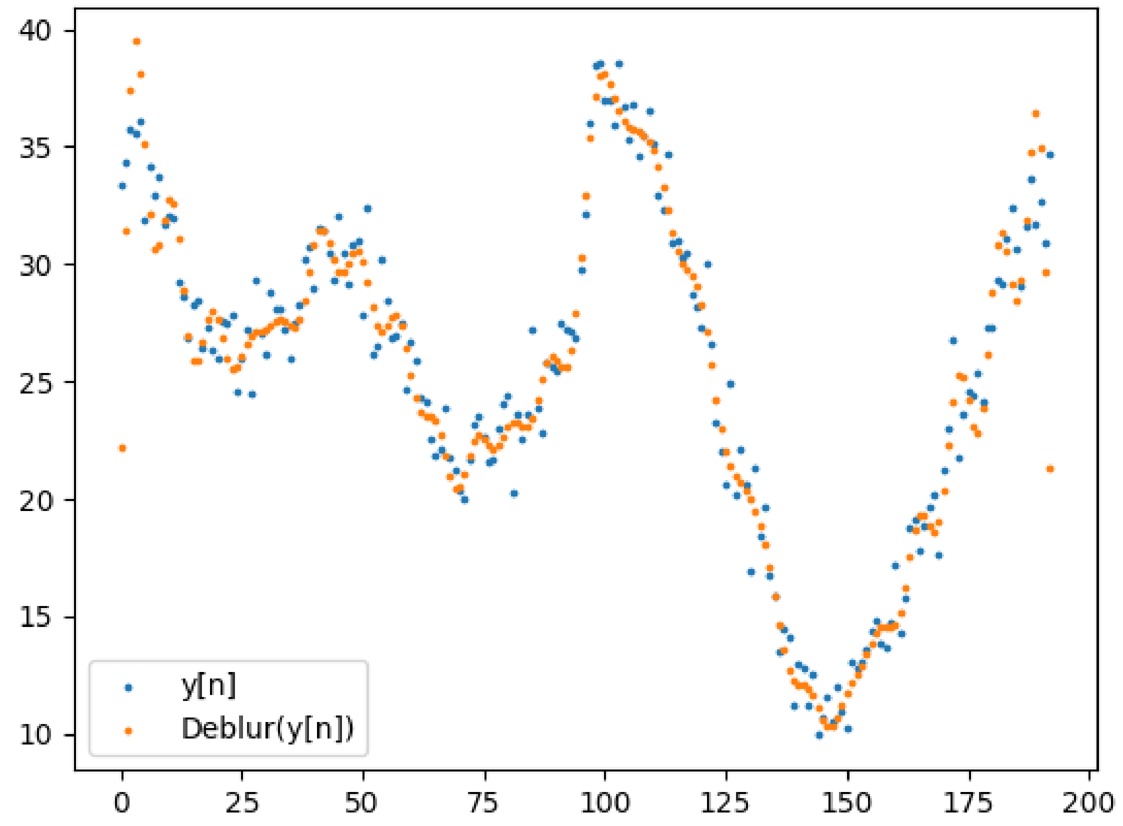
y[n] vs Denoised y[n]

Deblurring

For Deblurring we first took the Discrete Time Fourier Transformation of y[n] and h[n]. Then we took the Inverse Discrete Time Fourier Transformation of F(y)/F(h).

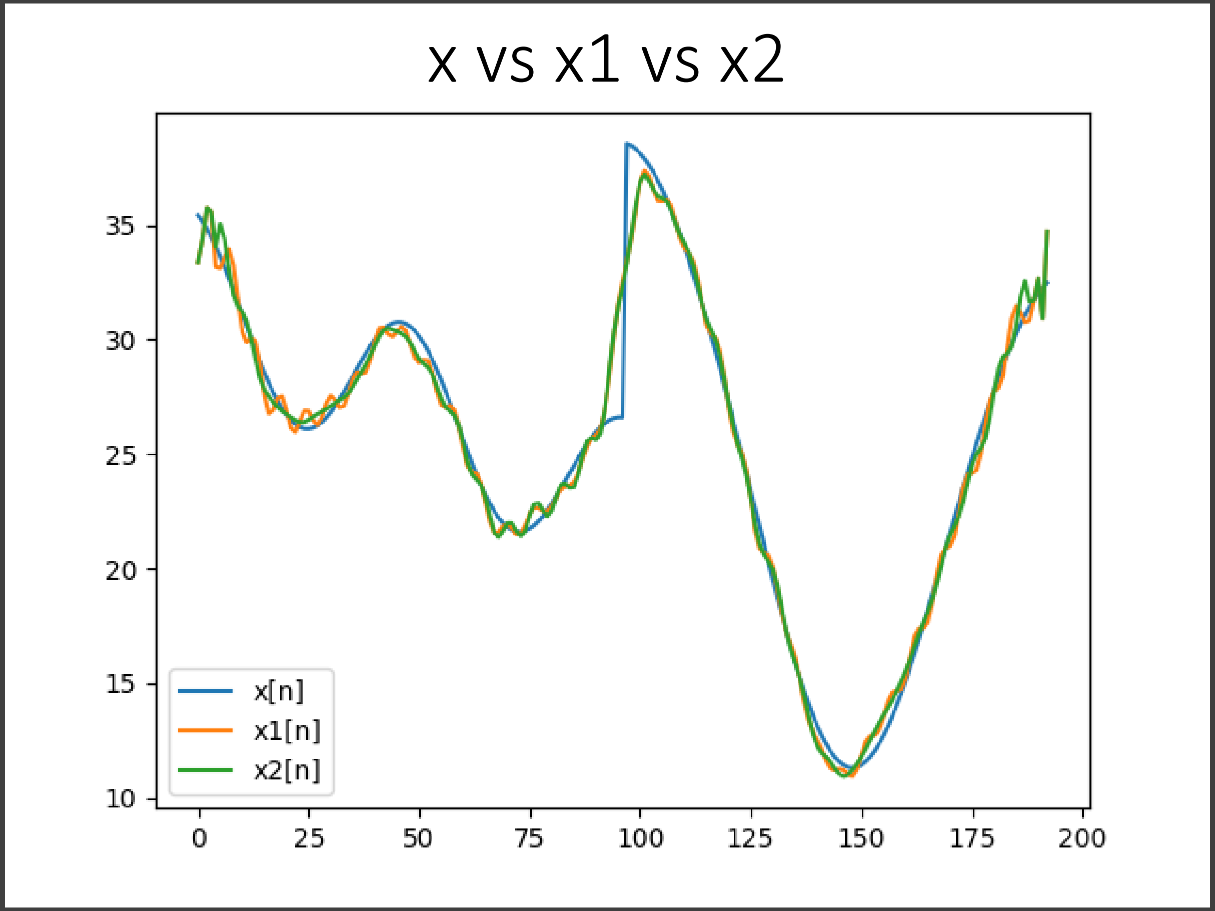
Fourier Transformation helps break the signal into its sin and cos components. This way knowing the Impulse Response used for blurring, we can reverse the effect (deconvolution).

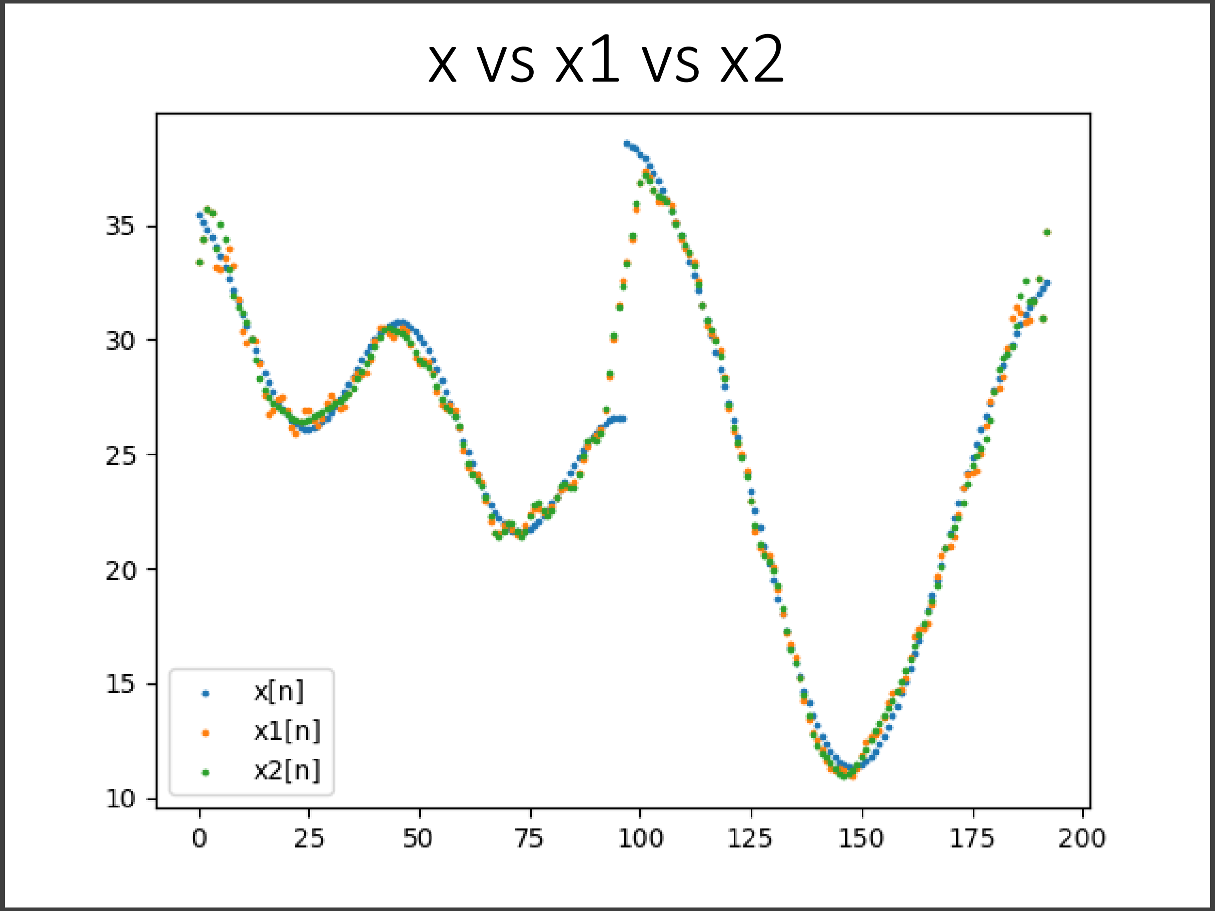
For DTFT we took angular frequency from 0 to 2pi with interval of 0.001 (Lesser the interval, lesser the error). Reason for this was that DTFT is periodic with period of 2pi and for IDTFT we need integral for an interval of 2pi only.



y[n] vs Deblurred y[n]

Observation





x1 : First Denoise then Deblur
x2 : First Deblur then Denoise
We can notice that we were able to recover most of the part.
The Impulse Response was low which caused the output to get a bit unstable.

Due to boundary effect from denoising and deblurring we replaced first 4 and last 4 elements of x1[n] and x2[n] with y[n]

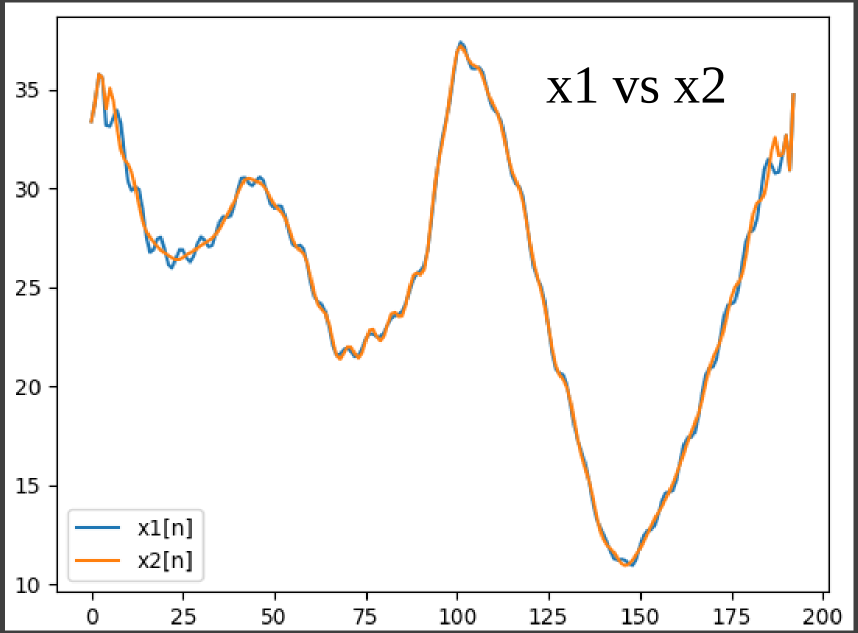
Conclusion

From the graphs we can observe that x1 and x2 are quite similar.
[The reason behind this is that the noise in the temperature values is quite high and the filter (Impulse Response) didn’t do much]

But it can be observed that x2 is slightly better than x1
Standard Deviation (x,x1) : 1.00
Standard Deviation (x,x2) : 0.95

The reason behind this is that in the original signal first noise must have been added and then it must have been blurred

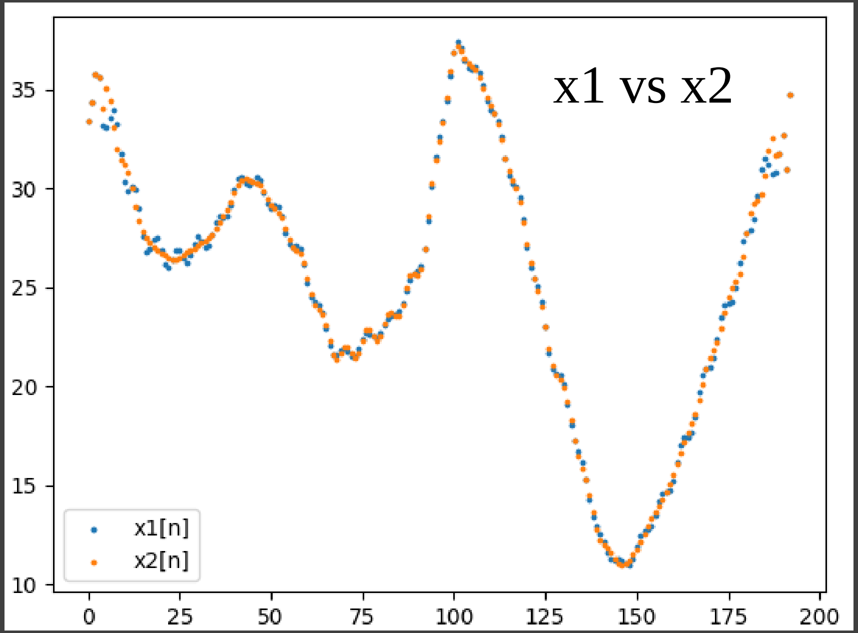
From this observation we can formulate an equation between x[n] and y[n]



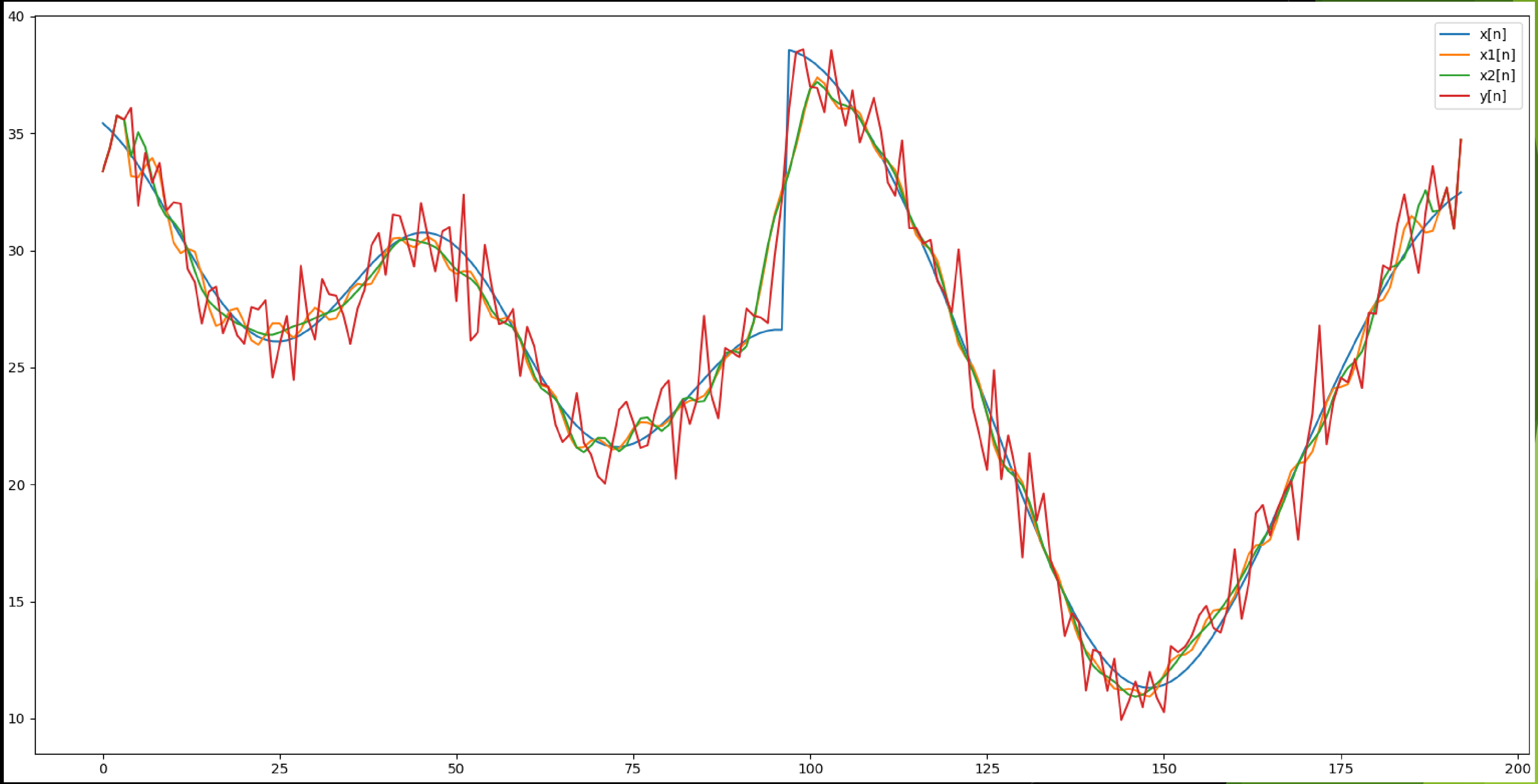
Equation : $y = (x + Noise).h$

$$x1 = \frac{y - N}{h}$$
$$x1 = \frac{(x + N).h - N}{h}$$

$$x2 = \frac{y}{h} - \mathrm{N}$$
$$x2 = \frac{(x + N).h}{h} - \mathrm{N}$$



x[n] vs y[n] vs x1[n] vs x2[n]



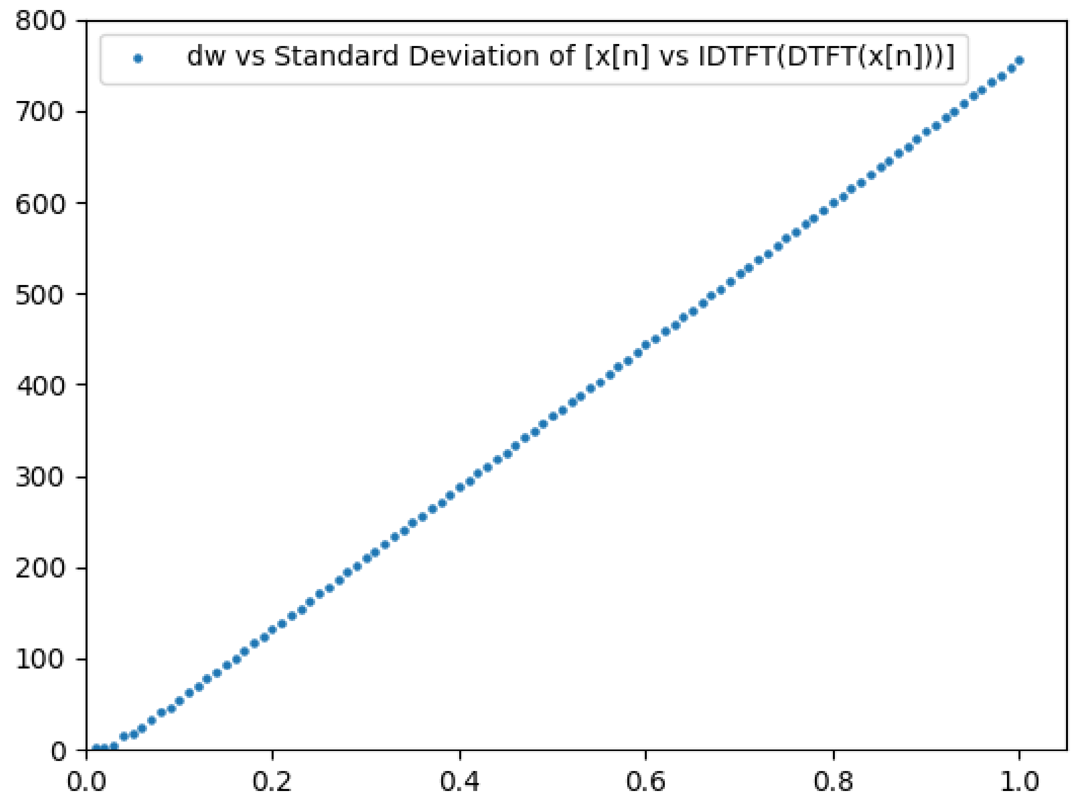
DTFT & IDTFT

We made a function DTFT for taking Fourier Transformation of the given Discrete Data. It is a periodic function with fundamental period of 2pi. We input dw in the function which is the interval between the x axis of each point in the DTFT. For dw tending to 0, DTFT will tend to being continuous.

We made a function IDTFT used for taking inverse Fourier Transformation. In a way it is Riemann Integration of the DTFT with interval of dw.

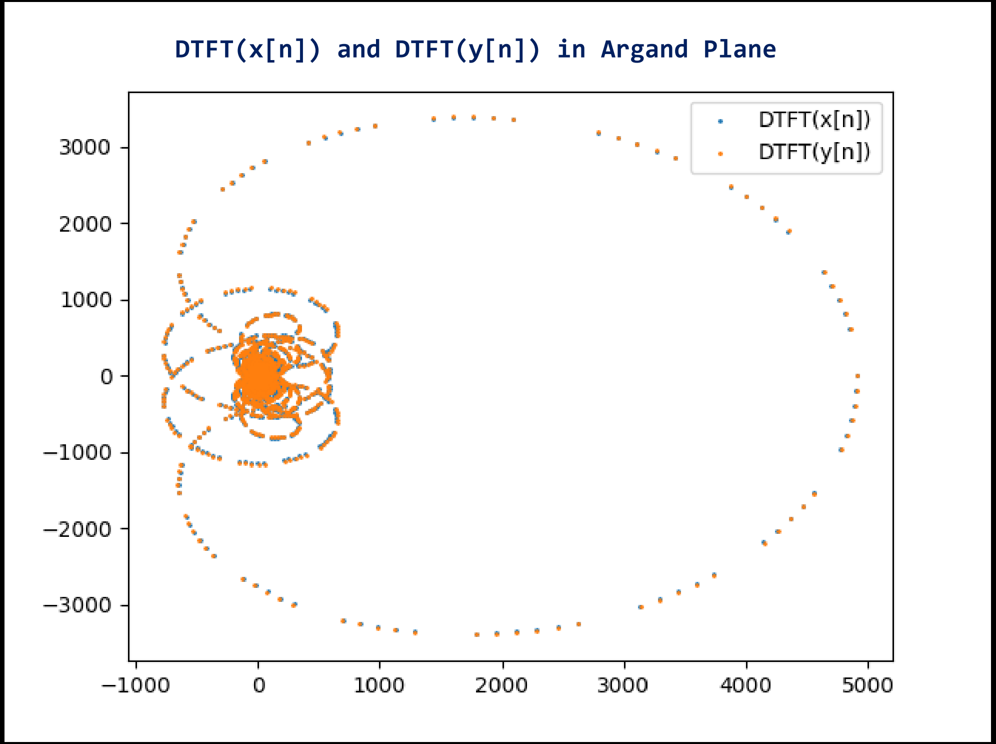
For better results dw needs to be small as lesser the dw, lesser the error caused due to discreteness.

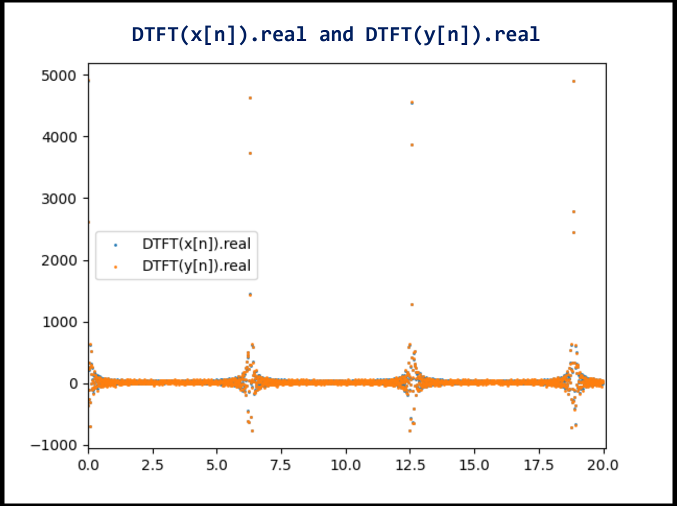
Note : Lowering the dw increases the computation time as the number of data points increases

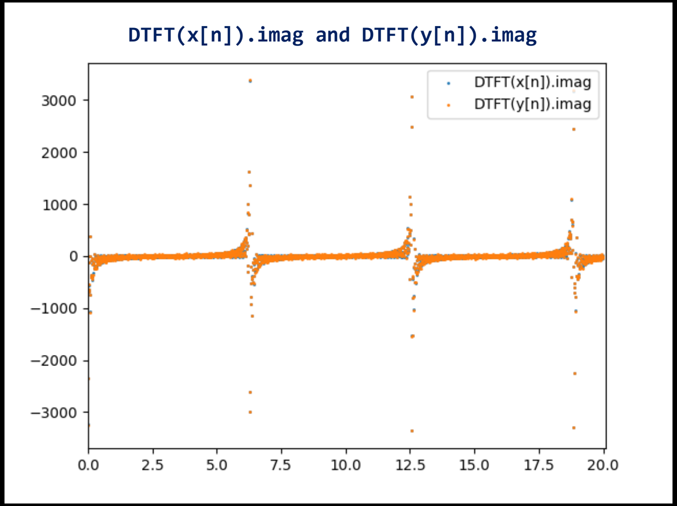


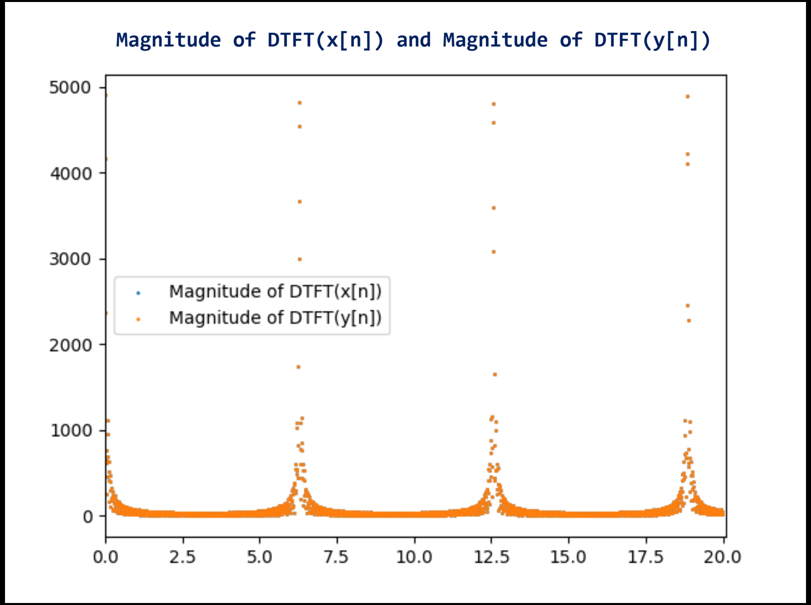
Plot to visualize how increasing the dw increases the error

Visualizing DTFT(x[n]) and DTFT(y[n])

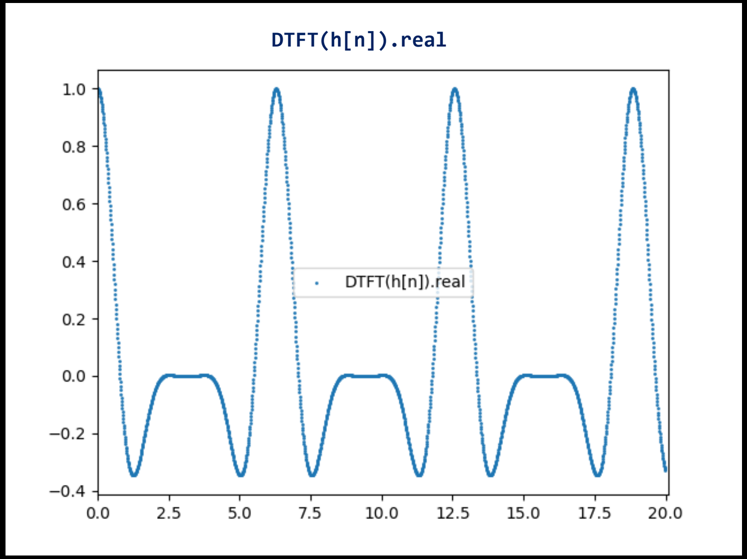


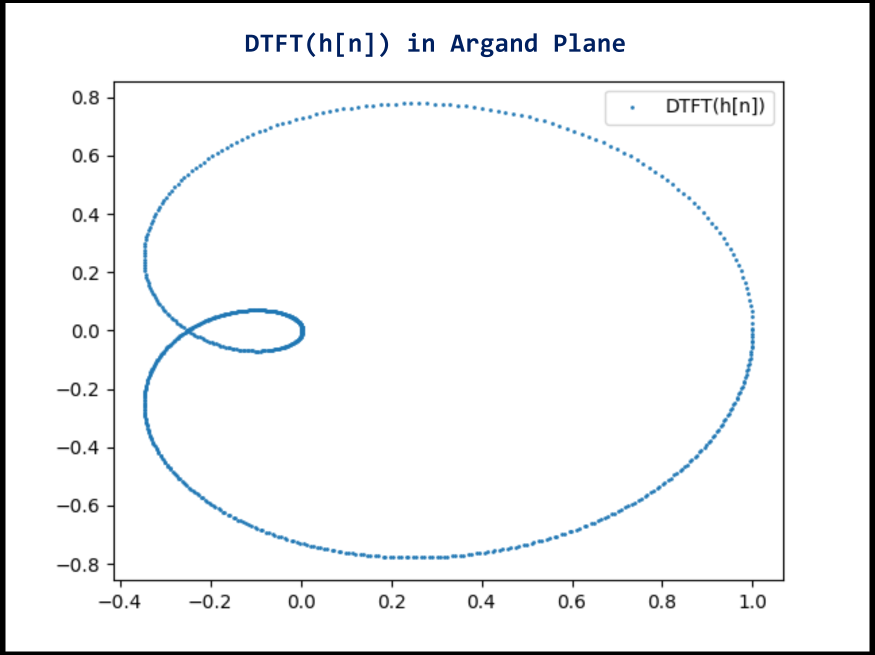


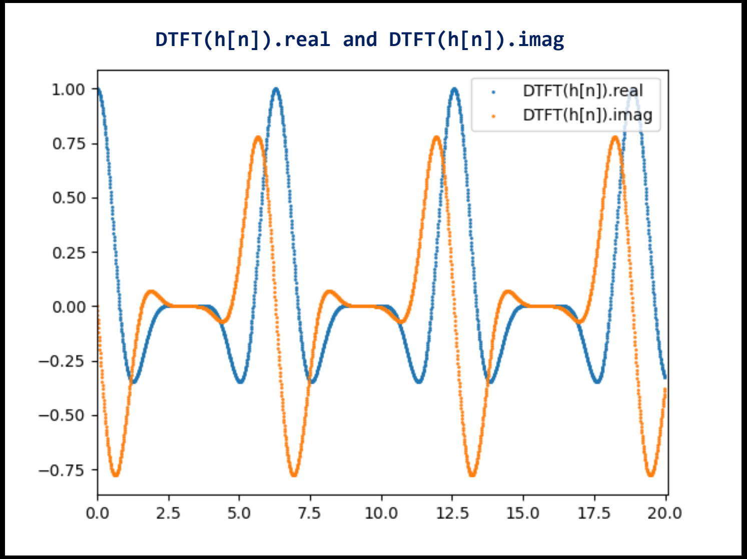


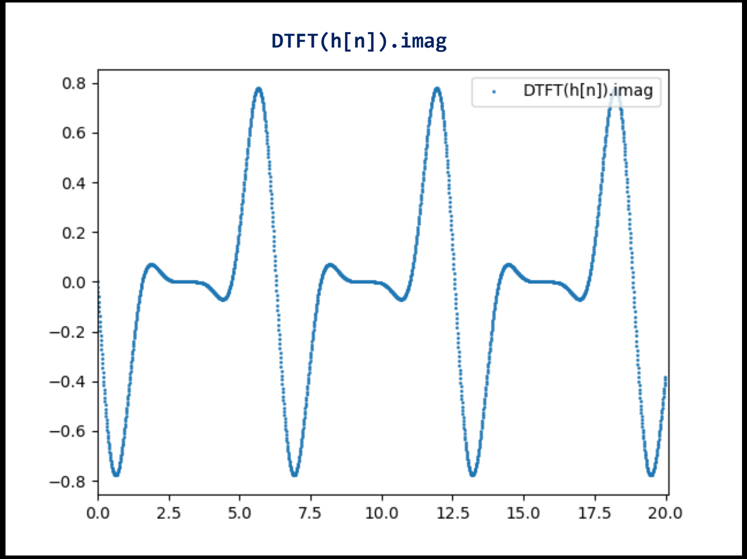


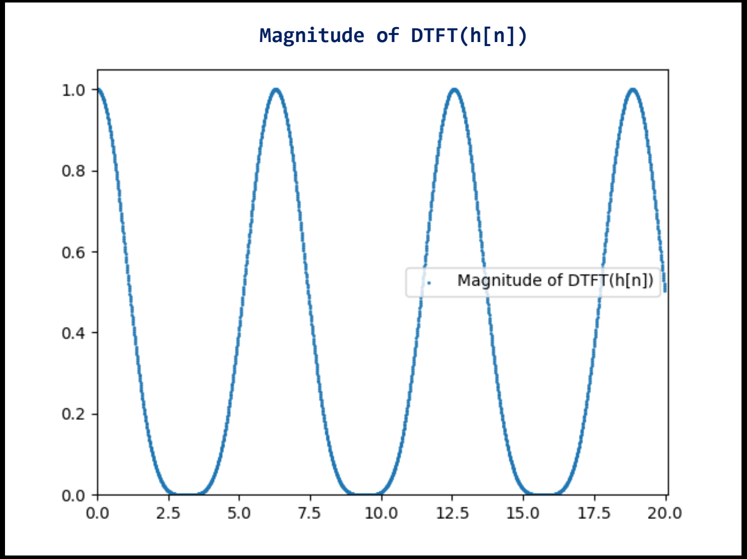
Visualizing DTFT(h[n])


DTFT(h[n]).real
DTFT(h[n]).real


DTFT(h[n]) in Argand Plane
DTFT(h[n])


DTFT(h[n]).real and DTFT(h[n]).imag
DTFT(h[n]).real
DTFT(h[n]).imag


DTFT(h[n]).imag
DTFT(h[n]).imag


Magnitude of DTFT(h[n])
Magnitude of DTFT(h[n])